11/11/13

**ssh c53.utdallas.edu**

*putty.exe*

```
login c51 or c52
```

& then c ssh to c53

---

**c53.utdallas.edu**

**Deadlock Avoidance**

**Deadlock Recovery**

Find a *deadlock breaking set.*

Pi... Pn are in a deadlock.

*terminate Pi [abnormally]*

when Pi terminates, all resources held by Pi are released.
Deadlock breaking set is a subset of deadlocked processes whose abnormal termination will lead to other processes to complete normally.

$$p_1, c_1, p_2, c_2, c_i = \text{Cost of terminating}$$

$$p_i, c_i$$

Optimal Deadlock Breaking Set:
Deadlock breaking set of smallest cost.

NP-Complete problem

Special cases where deadlocks do not arise.

Read book.

END OF DEADLOCKS
Memory Management

Instruction Execution Cycle

- Fetch instruction
- Decode
- Fetch operands (if needed) from RAM
- Execute
- Save result in memory if needed.

Compiling, Linking, Loading & Executing

Symbolic Addresses

Programs in High Level Langs. ⟷ Compiler ⟷ Object modules + info for linker + Relocatable Addresses ↓ Load Module

Object modules ↓ Link Editor

What type of addresses?
0 = offset
= 16

Module A

int c = 0;

variable c is allocated
space here

All references to var c
will be (A, 16)

Module offset

Addresses must be converted &
translated.

1. Symbolic addresses [i]
   Base Register

2. Relocatable address [(B, offset)]

3. Logical or effective address

4. Absolute address: we give to
   Memory Controller.

---

Simple Memory Management:

Fence

Register

OS

user

program
cannot access
OS part of RAM

RAM
Static Partitioned Allocation

# of partitions & Size of individual partitions are fixed

Data Structure?
Array or Table?

<table>
<thead>
<tr>
<th>#</th>
<th>SA</th>
<th>Size</th>
<th>Status</th>
</tr>
</thead>
</table>

Starting Address | Size | in use or Free?

Allocation: Look for smallest free partition of sufficient size
Deallocation: Change status to Free

Poor memory utilization

Fix? Use Dynamic Partitioned Allocation
III Dynamic Partitioned Allocation

Initially

When a process arrives, allocate needed RAM from FREE memory and rest is free memory.

Data structure: use linked list.

Inside OS:

Allocation? ? < First Fit
Best Fit
Worst Fit

Deallocation?
First fit: find 1st free partition whose size $\geq$ required Prog size

Best fit: 

Find smallest free partition of sufficient size (whose size $\geq$ Prog size)

Fragmentation: External

Have lots of free memory but in large # of small chunks.

In terms which of the 2 allocation methods is good from fragmentaion point of view? at all times?
Neither is a clear winner at all times.

Example 1

Requests: 48, 54, 26, 26, 46, 27, 28, 45

3 free partitions.  First fit

| 48, 26, 26 | 48, 26 |
| 54, 46, 46 | 54, 26 |
| 27, 28, 45 | 46, 27 |

Example 2

Requests: 49, 52, 48, 51

First fit

| 49, 48 | 49, 51 |
| 51, not allocated |
| 52, 48 |
Compaction? Defrag?

Optimal Compaction: NP-Complete.

Segmentation.

Sizes of
the functions

Segment 1
Segment 2
Segment 3
Segment 4
Segment 5
Segment 6

Program

Function 4

B

C

D

E

Request free memory of size S units

\[ S = \text{Size} = S_1 + S_2 + S_3 + S_4 + S_5 + S_6 \]

Addressing: \((\text{segment #, offset})\)

Effective addresses

Each segment is allocated separately

Segment table is needed to convert effective addresses into absolute addresses
All addresses in the user program will be \((1, xxx), or \((2, yyy)\)

\((5, 0) \rightarrow\) absolute address

is done as follows:

Consult a table (segment table)

Use \(S\) as an index & find

Starting address.  Add offset to starting address

L offset is \(\leq\) segment size

You get absolute address.

Segment table has 2 entries for each segment of the program.

Size:

This is Dynamic Address Translation
What about segmentation?

Overheads:
1. Maintain segment table in PCB
2. Dynamic Address Translation

Program

| main |
| function A |
| B |
| C |
| D |

Same Program

effective address \rightarrow (s, 0)

Sat 9:30-10:30